Hand Gesture Recognition system using FPGA

By

ASC Muhammad Umar Farooq

PC Muhammad Hamza khan

ASC Muhammad Hasnain

PC Shan

Supervised by:

Dr. Abdul Wakeel

Submitted to the faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad,

in partial fulfillment for the requirements of B.E Degree in Electrical (Telecom) Engineering.

JUNE 2022

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

“Hand Gesture Recognition system using FPGA”

is carried out by

Muhammad Umar Farooq

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical (Telecom.) Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr. Abdul Wakeel Ph.D

Department of EE, MCS

Date: ____________

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

The accomplishment of this project would have been possible without the contribution of the different people the names of which cannot be reckoned with. Their assistances are sincerely respected and thankfully recognized. We would like to thank following.

- ❖ To Allah Almighty who is most Powerful and Merciful and everything in in obedience.
- ❖ Our supervisor Dr Abdul Wakeel (PhD) for his support and guidance during the entire project.
- ❖ We acknowledge my sincere ineptness and gratitude to my parents for their love, dream and sacrifice throughout my life. We are thankful for the sacrifice, patience and understanding that were inevitable to make it possible. Their sacrifice had inspired me from the day we learned how to read and write until what I have become now. We cannot find the appropriate words that could properly describe my appraise words that could properly describe my appreciation for their devotion, support and faith in my ability to achieve my dreams.
- ❖ Lastly, we would like to thank any person who contributed to my final year project directly or indirectly. We would like to acknowledge their comments and suggestions, which was crucial for the successful completion of this study.

Plagiarism Certificate (Turnitin Report)

This thesis has_______11%___similarity index. Turnitin report endorsed by Supervisor is attached.

Muhammad Umar Farooq
00000278797

Muhammad Hamza khan
00000278785

Muhammad Hasnain
00000280706

Shan
00000278792

Signature of Supervisor

Abstract

Convolutional Neural Network (CNN) is an in-depth learning algorithm widely use in image processing and pattern recognition due to its robustness that includes flexibility. However, it is also the most calculated that leads to real-time malpractice. Field Programmable Gate Arrays (FPGA) works well for power consumption, flexible processing flexibility and pipeline performance. It is therefore expected to be used to accelerate an in-depth learning algorithm. In this study, a FPGA-based system was developed to detect hand-held visual perception in real time. We train the CNN model caffeinated model and get the model parameters on the PC. The Bilinear interpolation algorithm is used to adjust the image size taken by the camera. We then use FPGA to perform the Guessing Hand Gesture Recognition process and the parameters obtained by designing the accelerator using Xilinx tools.

Table of Contents

List of Figures

Chapter 1

Introduction

A person who is unable to express themselves verbally can make good use of hand gestures to get their message across. User-friendly communication is enhanced by the use of gestures in conjunction with speech language and facial expressions. In addition to visual games, security, and sign language identification, there are numerous other applications of human hand gestures based on a personal computer interface. When people interact with computers using a human hand gesture, a real-time system for recognizing and distinguishing various hand gestures is needed. There is still a long way to go in identifying the numerous hand motions provided by various academics. It takes a lot of training data to implement the recommended solutions, which in turn slows down the processing time and limits their application in real-time systems Real-time recognition of human hand movements is possible because to a new hand gesture recognition technology that addresses the issues raised above. Hand motions can be detected using simple shape-based features calculated by the proposed approach. FPGA implementation is also fast enough for real-time human- computer interaction because of the little processing time required by the FPGA software.

Using varying background illumination conditions and lighting adjustment, the suggested technique improves accuracy. The skin color separation is also employed for hand separation in order to limit the likelihood of other items in the background being falsely separated.

1.1 Problem statement

It is common practice in sign language to employ hand gestures. Voice-impaired people find it extremely challenging to communicate with the globe.

Hand gestures may be automatically recognized and converted into speech and text output by this project, making it easier for persons with disabilities to communicate with others.

1.2 Project description

There are three main processes in from data set creation to image processing and gesture recognition and classification. The method starts when image sensor captures the image. The primary thing after image capture is pre-processing. FPGA takes pictures in HEX form, but our picture is in binary form in 0 & 1 bits because it came from a camera. So, we do pre-process in MATLAB and convert out picture in to HEX form. After the pre-processing comes the post-processing. In post processing we do some basic image processing to extract the features of the image i.e. thresh-holding and border determination etc. The last but not least is that the Deep learning model. During this stage we take the method images and their feature and train our machine to be ready to identify and make a conclusion. Image is simply captured by a tool and is in digital form.

1.3 Prospective Application Areas

- It is utilized by vocally impaired person to speak with the globe in real unit of time.
- It is installed/use in schools for differently abled person.
- In home, offices and public places, a vocally person can communicate easily.
- To cover a huge gap between the globe and also the vocally impaired person.

1.4 Scope, objectives, and specifications

1.5.1 Scope and objectives

- Improve the system's capacity to recognize different types of lightning.
- Increasing precision by using a greater variety of gestures.
- Make a list of as many gestures as you can.
- This involves using gesture recognition to open web applications.
- Using gestures create a system for modifying.

1.5.2 Specifications

1.5.2.1 Hardware specifications

1. FPGA Spartan 3E family
2. PL2303 USB to TTL Converter Module
3. Webcam

1.5.2.2 Software specifications

1. Xilinx 14.7 version
2. MATLAB (2019 a)

1.5 Organization of the report

The report is consisted of five chapters. Chapter 2 briefly discussed the previously published work. Our proposed methodology with the working of different components presented in chapter 3. Chapter 4 shows and discusses the results. Conclusion and future work directions are discussed in Chapter 5.

Chapter 2

Literature Review

A literature review is a necessary step in developing a concept for a brand-new project. The features of a previously introduced similar project are modified and enhanced in an exceedingly new project. An in-depth study regarding all similar projects is additionally mandatory for the event and replacement of a project that uses FPGA for hand gesture recognition. Our research focuses on the subsequent points.

- Industrial Background
- Existing projects and their shortcomings
- Research Papers

2.1 Industrial background

In Pakistan out of the 23 Cr population, about 0.5083 Cr people, or 2.21% of the total population, are classified as disabled, with a speech disability accounting for 7.5% of this group.
Sadly, some persons are left unable to communicate after an accident.
Deaf people have relied on sign languages for generations to communicate with others, particularly those who are unable to speak or hear. Around the world, wherever there are people with disabilities, new sign languages are emerging. It is possible to represent a speaker's thoughts via sign language by combining hand forms, hands gestures, arms or body movements, and facial expressions. [1]

Normal people, on the other hand, don't appear to understand language communication. As a result, it may be difficult for persons who communicate solely through signing to speak or even

To explain their thoughts to others. Technology has made it possible for people with vocal impairments to communicate with others, therefore new solutions have been developed.

Thus, academics are working on a variety of approaches to convert hand motions into words. A vision-based system and a wearable device are the two most commonly used methods. " Images are processed using feature extraction techniques to detect hand and finger movements in vision-based systems. As detailed in, there are numerous other investigations into the use of vision-based systems for linguistic communication translation. Vision-based systems may not necessitate the use of sensory equipment that may be messy and uncomfortable for the users. Complex and intensive computations in designing algorithms for feature and movement recognition are required to develop vision-based systems.

For linguistic communication translation, wearable devices typically use sensors linked to the user or gloves. The devices are capable of converting words and phrases into text or even spoken language.

Hand gestures will be a really important thanks to communicate with computers. There are many potential application areas for hand gestures based. There are many difficulties related to the interaction of speechless people with normal people. Communication via visual aids is simpler than communication via verbal means. Sign language enables people to speak using anatomy language; each word contains a set of actions that represents a particular expression computer interface, including linguistic communication recognition.

Human computer interaction requires an individual's hand recognition system that may recognize and classify a spread of hand gestures in real time.

The project Hand gestures recognition system using FPGA reduces the communication gap between vocally impaired and normal persons.

2.2 Existing projects and their shortcomings

There are some different solutions being proposed for the hand gesture recognition system, but everyone has its own advantages and drawbacks.

Different solutions are preliminary being handed for the hand gestures recognition system, but every project has some advantages and shortcoming. Following are some results which are formerly being set and being applied. [2]

- Hand gesture recognition by using Raspberry pi.
- Recognition of hand gestures supported self-organization technique employing a Data Glove.
- Recognize the hand supported the form.
- Recognize hand gestures using contour analysis of hand.
- Hand gesture recognition by using fix threshold for hand segmentation.
- Pen based gesture recognition.
- Tracker-Based Gestures Recognition.
- Hand gesture recognition by using Arduino.

2.2.1 Hand gesture recognition by using Raspberry pi

The inability to talk is taken into account a real obstacle. People with this disability use other ways to speak with others. There are many various ways of communicating with them, one such common way of communicating is linguistic communication. Signing allows people to speak using organic structure language. Each word encompasses a set of human actions that represent a specific expression.

The motivation for this project is to translate human language into voice that understands human gestures. This can be achieved with the assistance of the Raspberry Pi web camera and speakers. Several linguistic communications to voice conversion systems are available, but none of them provide a transportable program. See if an individual with a voice disability can exchange front of the system and act, and also the system converts human gestures into voice and plays it aloud, allowing the person to truly communicate with the group. This method also helps visually and speech impaired people communicate with one another.

Drawback:

Raspberry Pi generally has slower processing speeds compared to the FPGA.

A Raspberry Pi isn't powerful enough to assist in handling multiple tasks.

2.2.1 Recognition of hand gestures supported self-organization technique employing a Data Glove.

The following procedure is applied to acknowledge hand-gestures supported self-organization by employing a Data Glove.

- Angles of cutlet joints are measured by a Data Glove at some time.
- Each gesture is segmented from a series of hand shapes.
- The number of snapshots in each sign is adjusted to acquire data of standard length.
- An input vector for self-organization is produce by joining an order of 10-spatial help- shape headings.
- Finally, grouping of KD and-gestures is completed activity by self-arrangement in accordance with their correspondences.

Recognition rates on self-organization technique employing a Data Glove.
range from 88.9% to 100% for training data and 86.7% to 95.0% for test data in real scenario.

Drawback:

Users must wear an information glove to segment their hands, limiting their freedom to communicate with their hands exposed.

2.2.2 Recognize the hand supported the form

At 45 dpi on a flatbed scanner, the scanner captures the subjects' proper hands in an unconstrained stance. Prior to any further processing, a fixed style is applied to the silhouettes of

Hand photographs, which incorporates both rotation and translation of the hand as a whole as well as each of the individual fingers.

The Harsdorf distance of the hand forms and the independent component feature of the hand silhouette photos were examined to see how they compare. For groups with at least 500 participants, the classification and verification results were found to be excellent, indicating that hand-based recognition could be a viable secure access control mechanism.

2.2.3 Recognize hand gestures using contour analysis of hand

This project uses real-time implementation to recognize hand features. Using the background subtraction method, three cameras and also the hand region are retrieved in order to get the most features for hand gesture identification, the longer distance from the final analysis and hence the hand contour must be calculated. For practical purposes, Y changes within the vertical contour image are used to calculate arm angle and finger positions. Using a data set of roughly 1800 photos, this approach performed quite well and was effectively applied.

Drawback:

When a project requires a lot of arithmetic, the system is generally tuned for real-time processing.

2.2.4 Hand gesture recognition by using fix threshold for hand segmentation

Shape-based hand motion recognition was employed in this study, although they used a set threshold for hand segmentation, which is susceptible to incorrectly segmenting other bright objects.

2.2.5 Pen based gesture recognition

It has been discussed a few times about recognizing motions from two-dimensional input devices like a pen or mouse. For example, the original Sketchpad system from 1963 featured light-pen motions. Since the 1970s, some commercial systems have employed pen gestures. Samples of gesture recognition include document editing, traffic control, and design jobs like splines.

Recent systems like the OGI Quick Set system have proven that pen-based gesture detection may also be used in conjunction with speech recognition in virtual environments. Pen gestures, including map icons, edit and taps are recognized by Quick Set. This includes 68 pen gestures. A variety of PDAs, including the 3Com Palm Pilot and a variety of Windows CE devices have been available for commercial use for many years now, beginning with the Apple Newton model. Gestural interfaces and interface designers are examined in depth by Long and Rowe in this paper.

2.2.6 Tracker-Based Gestures Recognition

There are many off-the-shelf tracking systems which will be used for gesture recognition, primarily to trace the road of sight, hand movements, and therefore the whole body and its position. Each sensor has its own strengths and weaknesses when interacting with a virtual environment. Since it may be useful to appear at the gesture interface, we are going to concentrate on gesture-based input by hand and body tracking.

2.2.7 Hand gesture recognition by using Arduino.

The Arduino-based project is functioning on a glove-based device accustomed convert linguistic communication (ASL) to speech. The essential system consists of two parts.

1. Sign language recognition, conversion to text

2. conversion to voice

Sign language gloves are equipped with a flex sensor and encompass simple gloves accustomed monitor finger flexion. Flex means to bend. These are sensors that change resistance in line with the flex of the sensor. The information from the sensor is shipped to the control unit, which is that the Arduino Nano. The analog signal from the sensor is digitally converted, compared to the stored character recognition value, and displayed as text on the LCD. Additionally, the text output is distributed wirelessly to a mobile or a PC consisting of test-to-speech conversion software. [3]

Limitations:

- An Arduino can only run one sketch or app at a time. Arduino doesn't support multitasking functionality.
- Lack of built-in peripherals
- Limited number of IDEs

Chapter 3

Gesture Recognition Process

There are three main processes in from data set creation to image processing and gesture recognition and classification. The method starts when image sensor captures the image. The primary thing after image capture is pre-processing. FPGA takes pictures in HEX form, but our picture is in binary form in 0 & 1 bits because it came from a photographic camera. So, we do pre-processing in MATLAB and convert out picture in to HEX form. After the pre-processing comes the post-processing. In post processing we do some basic image processing to extract the features of the image i.e. thresh-holding and border determination etc. The last but not least is that the Deep learning model. During this stage we take the method images and their feature and train our machine to be able to identify and make a conclusion.

Image is simply captured by a tool and is in digital form.

We discuss of these in details in follow.

3.1 : Pre-processing.

The first thing after image capture is pre-processing. We do the subsequent process on image.

3.1.1 : Image Re-sizing.

The first thing is image re-sizing. While re-sizing image we've got to kept some things within the mind that the larger the image is that the slow the general processing is and smaller the image the less he accuracy so we've got to create some balance in these thing. So, we resize our image in 96x72.

3.1.2 : HEX Conversion.

After re-sizing the binary image is converted into HEX from in order that we will deliver it to the FPGA for further processing. The MATLAB code does that conversion.

As soon as the FPGA has finished its post-processing, we need to import this image into the Deep learning model, and therefore we need to transform this image back to the binary format as well.

3.2 : Post-Processing.

Input to the FPGA is through port after HEX conversion. Within the 8-bit data format, it is loaded into the buffer (HEX data). After that, it's time to do some post-processing. In post-processing, the extraction of characteristics is completed.

3.2.1 : Grey scaling

Commencement of post processing is gray scaling. In FPGA we do gray scaling on the HEX type of pre-processed image.

Intensity information from the sun is all that each pixel's value represents. Typically, these photographs show only the blackest black to the whitest white. That is to say, the image is entirely composed of shades of grey, each of which has a different level of saturation.

3.2.2 : Filtering

We need to use some filter to detect the boundaries and fringe of the pre-processed picture so as to extract the most components and to urge the important details.

For the purpose of detecting edges, we employ the Sobel filter. Sobel filter has given us an accurate representation of the components, so now we'll proceed with further processing.

It is used in edge detection algorithms to create a picture with an emphasis on the edges of objects, and it is also known as the Sobel–Feldman operator or the Sobel filter.

Mathematical Formulation of the Sobel Operator

- **Gx = x-direction kernel * (3×3 portion of image A with (x,y) because the center cell)**
- **Gy = y-direction kernel * (3×3 portion of image A with (x,y) because the center cell)**
- **magnitude(G) = square root (Gx2 + Gy2)**
- **Ɵ = atan (Gy / Gx)**

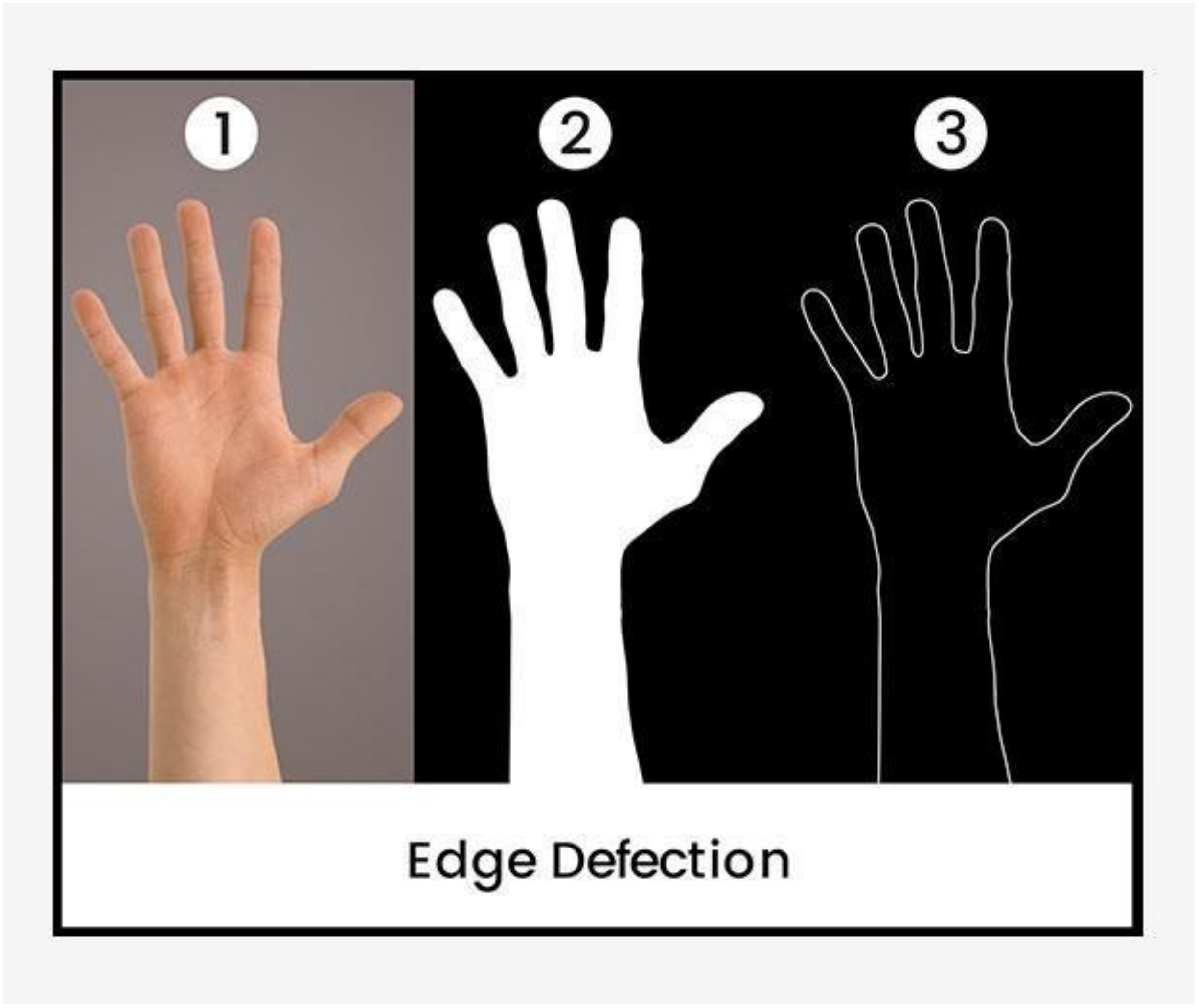


Figure 3.1: show a picture with edge detection and thresh-holding.

3.2.3 : Thresholding.

We apply global thresh-holding to the image to extract the ultimate component that's the gesture from the image.

After the thresh-holding we've our outcome that's absolutely the component of the gesture. Now we will use these components in our deep learning model to coach it and classify the results.

The data after pre-processing flow to the arrays where it stores a long time before it makes to MATLAB via interface then it converted back to the binary via HEX by MATLAB.

There are total 10 bits of knowledge first and also the last bit is start and end bits and other 8 bits are the information bits.

Here the post-processing is ended

Post processing figure:

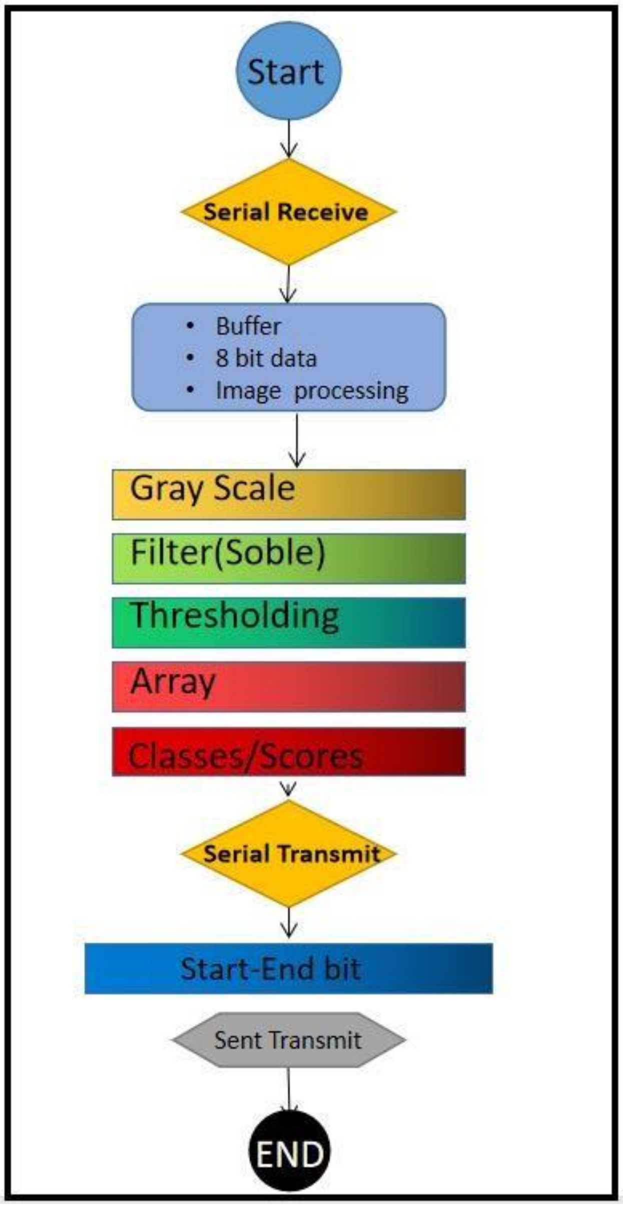


Figure 3.2: shows post processing diagram

3.3 : Deep Learning Model

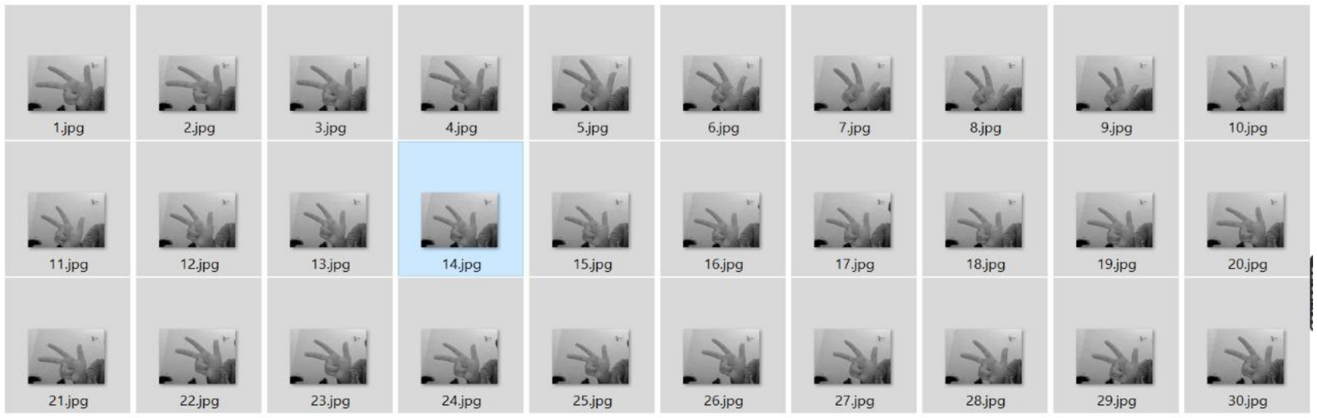
Deep learning is that the one amongst the foremost important step in Gesture recognition. This model is provoked after we are making the info set in the end the pre and post-processing steps the model is provoked to coach it first and so to classify the gestures. [4] [13]

3.3.1 : Convolutional Neural Network

CNN Convolutional Neural Network is the algorithm we're employing.
An algorithm known as a Convolutional Neural Network (ConvNet/CNN) may be able to learn from a picture by assigning significance to various aspects/objects in the image, and then being able to tell one from the other. The amount of pre-processing necessary for a ConvNet is significantly less than that required for other classification methods. In contrast to rudimentary approaches, Convolutional Neural Networks (ConvNets) can be trained to discover these filters/characteristics. New data sets are quickly optimized using ADAM's fast convergence. There are a variety of CNN designs that can be used to construct AI algorithms that can and will be used in the future. We're implementing Alex net's design in our project.

3.3.2 : Dataset creation

Once after we provoke this algorithm it seeks for its hunt for condition for dataset that if data set is already created and model is trained so it skips to classification except for the primary time as our data isn't created yet so it’s found the condition true.
First it creates the directories for all the ten classes (Gesture) and inside each class it creates 15 samples directories. After when directories are made it starts capturing the images its show the small print on screen that taking picture of sophistication 1, sample 1 and also the class 1 sample 2 so on. Every time when it takes the image, the image is pre-processed first so post-processed, and it saves into the respective directories that are made earlier.



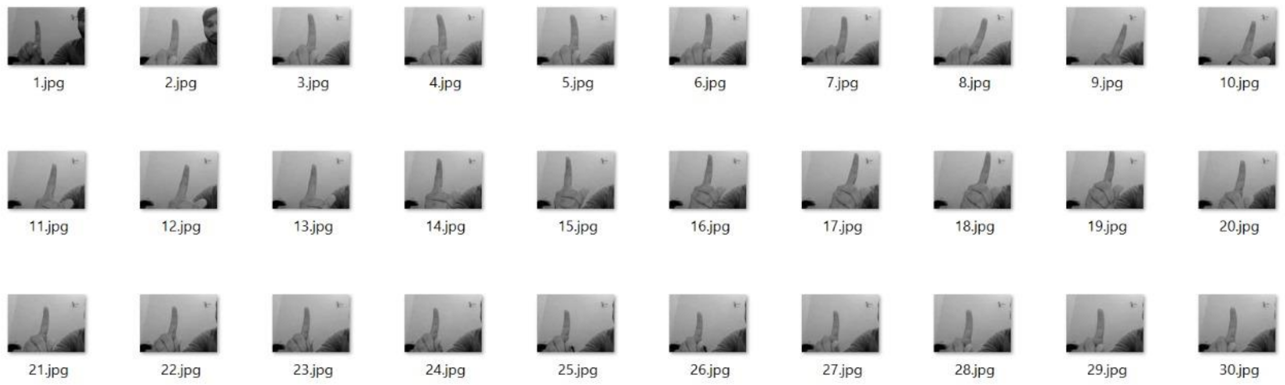
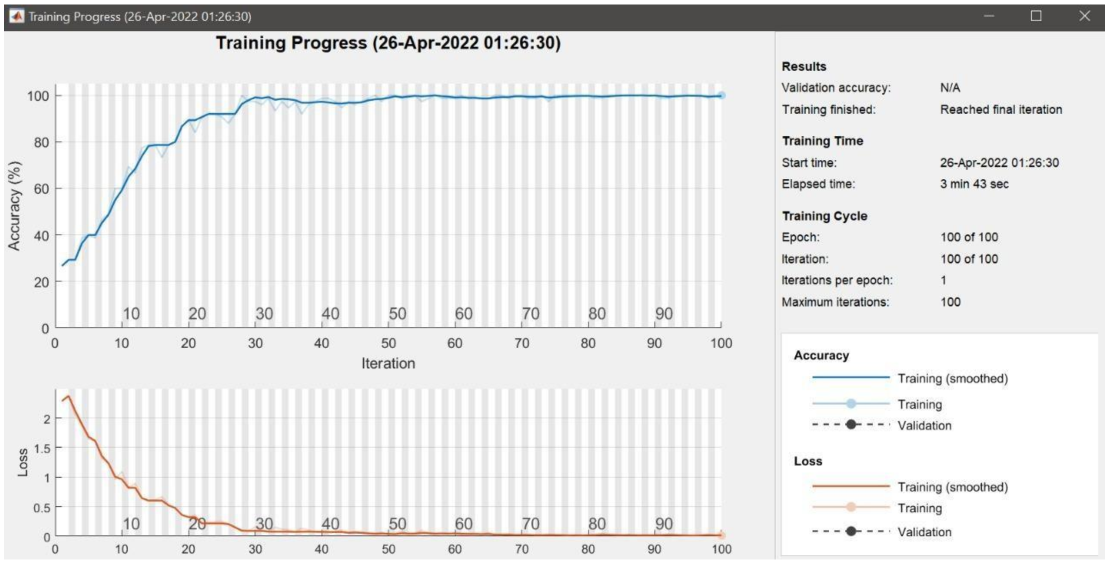


Figure 3.3 and 3.4 shows a gesture consisting of 30 samples

3.3.3 : Training the model

After the data set is created the data set condition is now false. So, it loads the model and starts the training phase where it trains the model. It is a time taking process and takes up to 5 to 6 hours to train. The more the sample, the more its take time but the more it is accurate.



Results
Validation accuracy: N/A
Training finished: Reached final iteration

Training Time
Start time: 26-Apr-2022 01:26:30
Elapsed time: 3 min 43 sec

Training Cycle
Epoch: 100 of 100
Iteration: 100 of 100
Iterations per epoch: 1
Maximum iterations: 100

Figure 3.5 show the training process of datasets

In figure 3.5 you can see the training process where accuracy is increasing, and loss is decreasing with no. of iteration. Also, we are taking 100 iterations.

3.3.4 : Classification of Gestures

The model is trained with our data set now. So, we are ready to use it. When we execute it, it finds these two conditions, either data set is created if no it creates one and above process repeated and if yes, its skip it and find the second, either.
Secondly it finds the model is trained or not if condition is yes, so it starts the training process and if not, it skipped to the final step that is classification.
The webcam is opened, and it captures the picture. Once the picture is captured there is going to be a pre- and post-processing and after this the processed image is taken to the deep learning

Model where it matches this picture with the data set and watch for the potential match. It measures how much the image is relevant to which class of the dataset. [7]

3.3.5 : Displaying the results

There are total 10 gestures, it classify the gesture only if it is matched more than 50% form the sample in the dataset. There is a sentence after each gesture. So, its display that sentence when a gesture is matched to the dataset.

3.3.6 : Text to speech conversion

There is sentences against each gesture s after the gesture recognition, the gesture is displayed as a sentence and then there is a MATLAB program defined that convert the sentence into the speech signal and we hear the sound of the desired gesture and sentence.

3.4 : Optimization process

The main issue we faced when we first run this it was the speed. It takes approximately 1.5 minutes to before its process the image and shows us the results. It was quite slow.

There were two reasons for this unnecessary delay.

- Hardware limitation
- Software limitation

We are using Spartan 3E FPGA for the image processing. This is quite complex task for an old model of FPGA such 3E. The laptop we are using for MATLAB algorithm such as HEX conversion and Deep learning is also not suitable. So we have these software restrictions.

We also have to do pre-processing (HEX CONVERSION) on the MATLAB before and after post-processing this is also an additional task to do. FPGA has this limitation it only accepts the data in the form of HEX. We can use a specific image sensor that directly converts the picture into the HEX form before post-processing but then too we need it after post-processing. [10]

The software limitation we have is the deep learning algorithm. We are using MATLAB for this purpose because we cannot use FPGA for deep learning purpose. So, this is an additional task for the MATLAB and laptop.

We have tried some things to optimize the performance. We have decreased the size of picture as less then we can. The cons of this are that our accuracy can be affected by it. We have discarded the 50% of weak neurons and just considered 50% strong neuron, it helps us a lot to decrease the processing time. We are increasing the no of samples so that we can increase or accuracy a bit. Before we took 5 sample for each gesture no, we are taking 15 samples for each gesture. So, we increased the accuracy significantly as well as the speed. [5]

3.5 : Proposed Block diagram

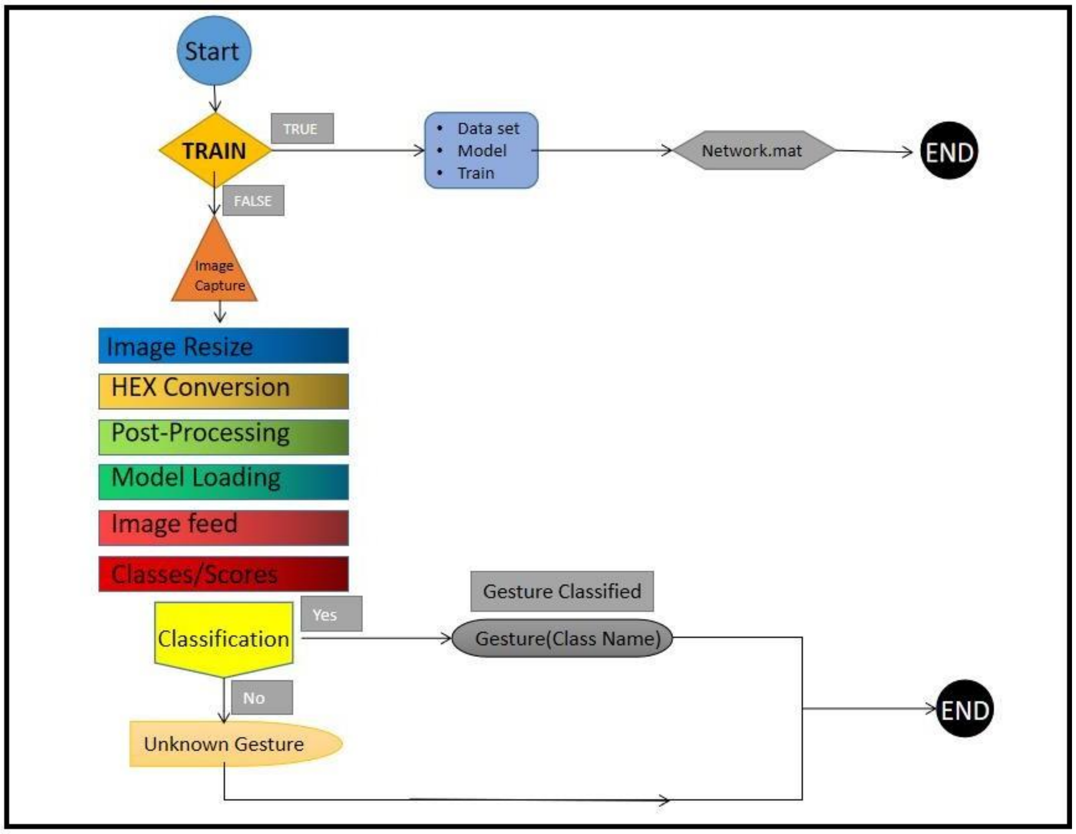


Figure 3.6 shows the proposed block diagram of Gesture Recognition Process

3.6 : Spartan-3E FPGA Family:

3.6.1 : Introduction:

High-volume, consumer-oriented electronic applications are the target market for the Spartan-3E family of Field-Programmable Gate Arrays (FPGAs) From 100,000 to 1.6 million system gates are provided by the five members of this family.

Spartan-3E improves on the previous Spartan-3 family's success by enhancing logic value per I / O and cutting expenses per sensible cell greatly. New features enhance system efficiency while lowering initial setup costs. To set new norms in formal reasoning, these Spartan-3E FPGA upgrades and 90 nm process technology deliver higher efficiency.

Consumer applications can benefit greatly from the Spartan-3E FPGA's low cost. The Spartan-3E series of ASICs is among the best kept secrets in the industry. FPGAs eliminate the high initial costs, extended growth cycles, and environmental instability of traditional ASICs. Additionally, unlike ASICs, the FPGA format enables for on-the-spot design development without the need for custom hardware modifications.

3.6.2 : Specifications:

• Logic Cells. Over 10,000.

• DDR SDRAM. 64 MByte, x16 data interface, 100+ MHz

• NOR Flash. 16 MByte (Intel Strata Flash)

• SPI Serial Flash. 16MBit (STMicro)

• Ethernet. 10/100 PHY. (Requires Ethernet MAC in FPGA)

• Oscillator. 50 MHz clock oscillator.

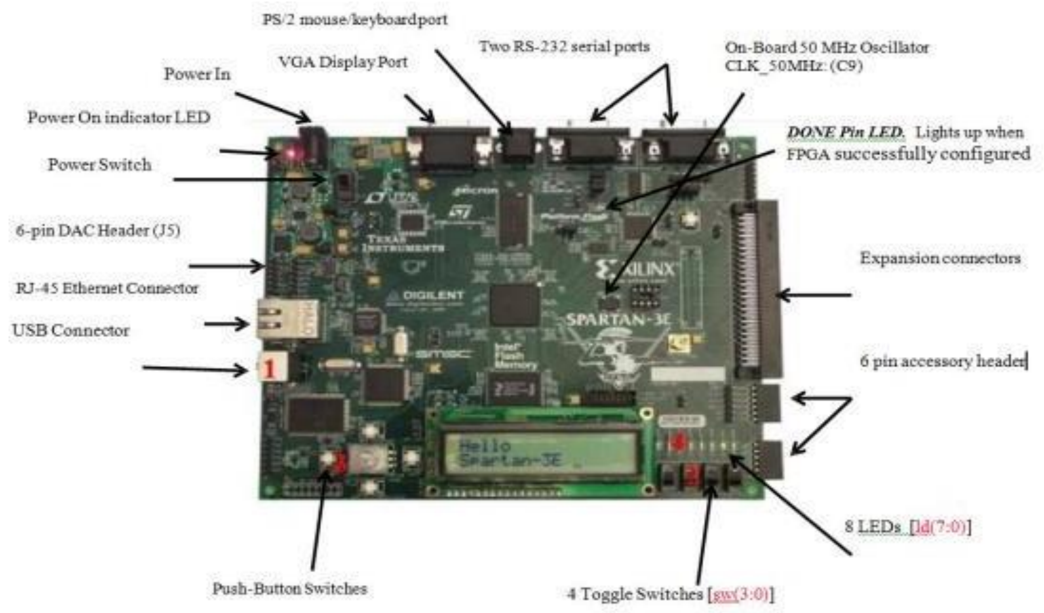


Fig 3.7 Spartan 3E descriptive diagram

3.6.3 : Functional Description:

An overview of the IOB to connect a packet pin to the FPGA's internal logic, an Input/Output Block (IOB) is used as a programmable unidirectional or bidirectional link. However, there are some key distinctions from the Spartan-3 IOB:

Only

• Added customizable entry delays for all blocks

For a limited subset of IOB capabilities, a unidirectional input-only block uses DDR flip flops that can be shared between neighboring

. As a result, it is impossible to find a way out of this situation. Any reference to output functionality is not valid for input-only blocks in the following paragraphs. Only the number of input blocks changes with device size, but never more than 25% of the total IOBs..

3.6.4 : Storage Element Functions:

All three pathways are represented by three pairs of storage elements in the IOB. Edge-triggered D-type flip-flops (FDs) and level-sensitive latches can also be used to implement these storage components. (LD). Double Data Rate (DDR) transmission can be generated from the output path or Tri-State path by using a dedicated multiplexer for this purpose. By turning the rising-edge synchronized data into rising-and-falling edge synchronized bits, this can be accomplished. Double Data Rate D-type flip-flops have two registers and a multiplexer (ODDR2). [6] [9]

3.6.5 : The Configurable Logic Blocks Overview:

Synchronous and combinatorial circuits can be implemented using CLBs, which are the primary logic source. Each CLB has four slices, each of which has two LUTs for logic implementation and two configurable storage items that can be used as flip-flops or latches. Additional multiplexers and transport logic simplify wide logic and arithmetic tasks, and LUTs are offered as 16x1 memory (RAM16) or 16-bit shift register (SRL16). CLB slices are automatically assigned to general-purpose logic in most designs. The Spartan-3E family CLB structure is identical to the Spartan-3 family CLB structure.

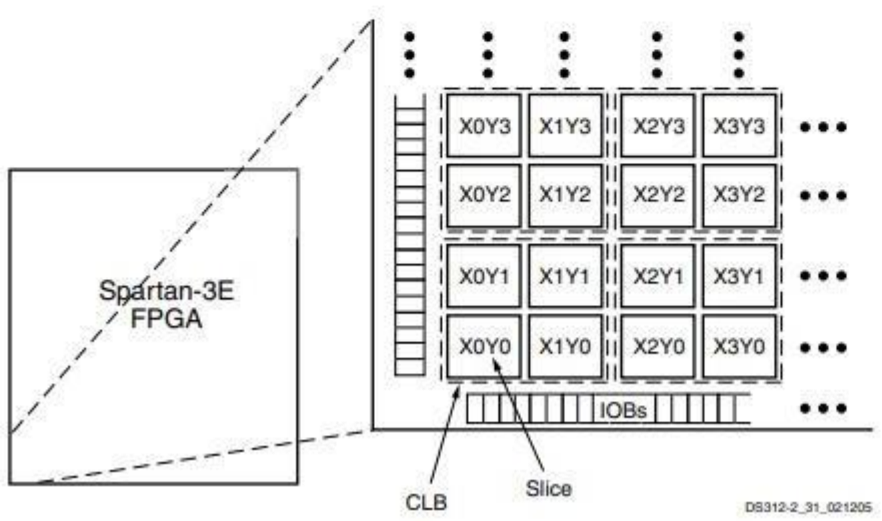


Fig 3.8 Configurable Logic Blocks

3.7 : Software Tool:

3.7.1 : Xilinx 14.7 version

For the synthesis and analysis of HDL designs, Xilinx ISE (Integrated Synthesis Environment) is Xilinx's discontinued software tool for the development of embedded firmware for Xilinx FPGA and integrated circuit (IC) product lines. Xilinx Vivado was the next product to come out. To run this project, Xilinx 14.7 is required.

3.7.2 : MATLAB 2019a:

High-performance language for technical computing, MATLAB is an excellent choice. You can solve problems by using conventional mathematical notation and see the results in a clear, easy-to-understand graphic. MATLAB 2019a is being used to collect the data set.

PL2303 USB to TTL Converter Module:

Incorporated with Prolific Technology's USB to UART bridge IC, the PL2303, the PL2303 UART Module provides a single-chip solution. Minimal components and board space are required for this USB-to-UART bridge controller to bring the RS-232 interface up to date with USB 2.0. Virtual COM Port (VCP) is a royalty-free virtual COM port that can be utilized in PC applications. The USB 2.0 controller, USB transceiver, and full modem control signal are all included in this module. [12] [14]

Features:

• Two data transfer indicators can monitor the data transfer status in real time

• Works with existing COM port PC applications

• Standard USB type a male and TTL 6 pin connector.

Pin Name	**Description**
3.3 V	3.3V Vcc pin
5.0 V	5.0V Vcc pin
TxD	Asynchronous data output (UART Transmit)
RxD	Asynchronous data input (UART Receive)
GND	Ground

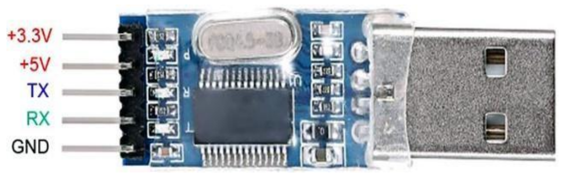


Figure 3.9 Pin out figure of USB to TTL converter

Chapter 4

Coding and Analysis

Creating dataset:

```
clc
clear all
close all
%% read images

if isfolder('Dataset')
    disp('path found!')
else
    disp('path not found!')
    disp('Creating path folders!')
    mkdir('Dataset')
end

%% create folders for dataset classes
classes= 5
samples = 15

cam = webcam();

for i=1:classes
    if ~isfolder(string(['Dataset' '\' char(num2str(i))]))
        mkdir(string(['Dataset' '\' char(num2str(i))]))
        for j=1:samples
            disp(['sample number: ' num2str(j)] )
            img = snapshot(cam);
            img = imresize(img(:,:,1),0.15);
            imwrite(img,string(['Dataset' '\' char(num2str(i)) '\' char(num2str(j)) '.jpg']));
            imshow(img)
            pause(1)
        end
    end
    disp('Capture data for new class')
    pause(5)
end
clear cam
```

Comparing dataset with real gestures:

```
clc
close all
clear all

train=false

if train

    imds = imageDatastore('Dataset',...
        'IncludeSubfolders',true,'FileExtensions','.jpg',...
        'LabelSource','foldernames');

    layers = [
        imageInputLayer([size(imread(cell2mat(imds.Files(1))))])

        convolution2dLayer(3,32,'Padding','same')
        batchNormalizationLayer
        reluLayer
        maxPooling2dLayer(2,'Stride',2)
        dropoutLayer(0.5)


        convolution2dLayer(3,64,'Padding','same')
        batchNormalizationLayer
        reluLayer
        maxPooling2dLayer(2,'Stride',2)
        dropoutLayer(0.5)

        convolution2dLayer(3,64,'Padding','same')
        batchNormalizationLayer
        reluLayer
        maxPooling2dLayer(2,'Stride',2)
        dropoutLayer(0.5)

        convolution2dLayer(3,96,'Padding','same')
        batchNormalizationLayer
        reluLayer
        dropoutLayer(0.5)

        fullyConnectedLayer(length(unique(imds.Labels)))
        softmaxLayer
        classificationLayer];



    opts = trainingOptions('adam', ...
        'MaxEpochs',100, ...
        'minibatchsize',128,...
        'Shuffle','every-epoch', ...
        'Plots','training-progress', ...
        'Verbose',false);

    net = trainNetwork(imds,layers,opts);

    save('network.mat','net');
else
    disp("taking image")
    pause(1)
    cam = webcam();
    img = snapshot(cam);
    img = imresize(img(:,:,1),0.15);
    size(img)
    clear cam
    figure,imshow(img)

    convertHex = reshape(img,1,[]);

k=1;
for i=1:length(convertHex)
a(k)=convertHex(1,i);
k=k+1;
end
fid = fopen('img.hex', 'wt');
fprintf(fid, '%x\n', a);
disp('Text file write done');disp(' ');
fclose(fid);
    %% perform serial communication link

    s = serial('COM5','BaudRate',115200,"Timeout",5)
    fopen(s)
%     fprintf(s,'RS232?')
    for i=1:length(convertHex)
        fprintf(s,char(a(i)))%,"uint8")
    end
    fclose(s)
    fid = fopen('img.hex', 'rt');
    fscanf(fid,'%f');
    disp('Text file reading done');
    fclose(fid);

    load network.mat

    [YPred scores]= classify(net,img)

    if length(find(scores>0.5))~=0
    classname = net.Layers(end).Classes(YPred)
    else
        disp('unknown gesture')

    end



end
```

FPGA coding:

```
[9:06 PM, 5/23/2022] ~Hasnain: clc
clear all
close all
%% read images

if isfolder('Dataset')
    disp('path found!')
else
    disp('path not found!')
    disp('Creating path folders!')
    mkdir('Dataset')
end

%% create folders for dataset classes
classes= 5
samples = 15

cam = webcam();

for i=1:classes
    if ~isfolder(string(['Dataset' '\' char(num2str(i))]))
        mkdir(string(['Dataset' '\' char(num2str(i))]))
        for j=1:samples
            disp(['sample number: ' num2str(j)] )
            img = snapshot(cam);
            img = imresize(img(:,:,1),0.15);
            imwrite(img,string(['Dataset' '\' char(num2str(i)) '\' char(num2str(j)) '.jpg']));
            imshow(img)
            pause(1)
        end
    end
    disp('Capture data for new class')
    pause(5)
end
clear cam
[9:11 PM, 5/23/2022] ~Hasnain: `timescale 1ns / 1ps



module main(
(*LOC = "C9" *) input clk_fpga,
(LOC = "D18") input reset,
(LOC = "D7") input RxD,
(LOC = "F9 E9 D11 C11 F11 E11 E12 F12") output [7:0]RxData,
(* LOC = "D16" *) output reg sf_e,
(* LOC = "M18" *) output reg e,
(* LOC = "L18" *) output reg rs,
(* LOC = "L17" *) output reg rw,
(* LOC = "M15" *) output reg d,
(* LOC = "P17" *) output reg c,
(* LOC = "R16" *) output reg b,
(* LOC = "R15" *) output reg a

);

reg shift;
reg state, nextstate;
reg [3:0] bit_counter;
reg [1:0] sample_counter;
reg [13:0] baudrate_counter;
reg [9:0] rxshift_reg;
reg clear_bitcounter,inc_bitcounter,inc_samplecounter,clear_samplecounter;

wire HCLK,HRESETn,VSYNC,HSYNC;
wire [7:0]  DATA_R0;
wire [7:0]  DATA_G0;
wire [7:0]  DATA_B0;
wire [7:0]  DATA_R1;
wire [7:0]  DATA_G1;
wire [7:0]  DATA_B1;
wire ctrl_done;

parameter clk_freq = 50_000_000;
parameter baud_rate = 9_600;
//parameter baud_rate = 4_800;
parameter div_sample = 4;
parameter div_counter = clk_freq/(baud_rate*div_sample);
parameter mid_sample = (div_sample/2);
parameter div_bit = 10;

assign RxData = rxshift_reg[8:1];

always @ (posedge clk_fpga)
        begin
                if (reset) begin
                        state<=0;
                        bit_counter<=0;
                        baudrate_counter<=0;
                        sample_counter<=0;
                end else begin
```

```
                    baudrate_counter<=baudrate_counter + 1;
                    if (baudrate_counter >= div_counter-1) begin
                        baudrate_counter <=0;
                        state<=nextstate;
                        if (shift) rxshift_reg <= {RxD,rxshift_reg[9:1]};
                        if (clear_samplecounter) sample_counter<=0;
                        if (inc_samplecounter) sample_counter <=sample_counter+1;
                        if (clear_bitcounter) bit_counter<=0;
                        if (inc_bitcounter) bit_counter<=bit_counter+1;
                    end
                end
            end


always @ (posedge clk_fpga)
begin
            shift <=0;
            clear_samplecounter<=0;
            inc_samplecounter<=0;
            clear_bitcounter<=0;
            inc_bitcounter<=0;
            nextstate<=0;
            case(state)
                0: begin
                    if (RxD)
                        begin
                        nextstate<=0;
                        end
                    else begin
                        nextstate<=1;
                        clear_bitcounter <=1;
                        clear_samplecounter<=1;
                    end
                end
                1: begin
```

```
                    nextstate<=1;
                    if (sample_counter==mid_sample-1) shift<=1;
                            if (sample_counter==div_sample -1) begin
                                    if (bit_counter==div_bit-1) begin
                            nextstate<=0;
                            end
                            inc_bitcounter<=1;
                    end else inc_samplecounter<=1;
            end

            default:nextstate<=0;
        endcase
end

                   f
                   r
reg [26:0] count   e
= 0;               s
r                  h
e                  ;
g                  always @
[                          (posedg
5                          e
:                          clk_fpg
0                          a) begin
]                          count<=
c                          count+1
o                          ;
d                          case (count[26:21])
e               0:code<=6'h03;
;               1:code<=6'h03;
r               2:code<=6'h03;
e               3:code<=6'h02;
g               4:code<=6'h02;
r               5:code<=6'h08;
e               6:code<=6'h00;
```

```
7                    :code<=6'h0C;
:                    10:code<=6'h00;
c                    11:code<=6'h01;
o                    12:code<=6'h24;
d                    13:code14:code<=6'h
e                    26; 15:code<=6'h25;
<                    16:code<=6'h26;
=                    17:code<=6'h2C;
6                    18:code<=6'h26;
'                    19:code<=6'h2C;
h                    20:code<=6'h26;
0                    21:code<=6'h2F;
6                    22:code<=6'h22;
;                    23:code<=6'h2C;
8                    24:code<=6'h001100;
:                    25:code<=6'h000000;
c                    26:code<=6'h25;
o                    27:code<=6'h27;
d                    28:code<=6'h26;
e                    29:code<=6'h2F;
<                    30:code<=6'h27;
=                    31:code<=6'h22;
6                    32:code<=6'h26;
'                    33:code<=6'h2C;
h                    34:code<=6'h26;
0                    35:code<=6'h24;
0                    36:code<=6'h22;
;                    37:code<=6'h21;
9                    default: code<=6'h10;

            refresh<=count[20];
            sf_e<=1;
            {e,rs,rw,d,c,b,a} <= {refresh,code};
    end

    //imageProcessingAlgorithm A1(HCLK,HRESETn,VSYNC,
    HSYNC,DATA_R0,DATA_G0,DATA_B0,DATA_R1,DATA_G1,DATA_B1,ctrl_done);
                          serialTransmitter
                          T1(clk,data,trans
                          mit,reset,TxD)
                          <=6'h28;

endcase
endmodule
```

Chapter 5

Conclusions

Our supervisor and us came up with the concept of using this method in conjunction with a neural network.
Because the network output can be used to make inferences, neural networks are useful. We can check the result and figure out the answer if the vector isn't correctly classified.
Optimization of the orienting algorithm is a possibility. It's important to be able to distinguish yourself from the crowd. It's obvious that the photographs have a role in this, but the algorithm also has a role to play. Cutting-edge line-detection algorithms are constantly being developed. Even though tangents are a new thought that has come to mind, there isn't enough time to pursue them now that they have crossed my mind. To state that I've reached solid conclusions at the end of the endeavor isn't a safe bet. Only the first phase of the project can be done this way. The output vector will always be the same, no matter how many times you execute the programmed. No such issue arises with perceptron neural networks. Apart from not being 100 percent stable, there are so many factors (e.g. number of layers, number of nodes) that one can experiment with that it's not that straightforward to discover the best settings. It's all about the application, as we've already stated. To meet a given noise target, for instance, you can try to meet these requirements. Among my primary objectives were speed and the absence of specific gear. It worked, despite the difficulties.

Although it would run quicker in C/C+, the concept and implementation would have been far more difficult. MATLAB is slower, but it allows its users to work faster and focus on the results rather than the design

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, p. 436, 2015.

[2] S.-H. Han and K.-Y. Lee, "Implemetation of image classification cnn using multi thread gpu," in SoC Design Conference (ISOCC), 2017 International. IEEE, 2017, pp. 296–297.

[3] S. I. Baykal, D. Bulut, and O. K. Sahingoz, "Comparing deep learning performance on bigdata by using cpus and gpus," in 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT). IEEE, 2018, pp. 1–6.

[4] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung, "Toward accelerating deep learning at scale using specialized hardware in the datacenter," in Hot Chips 27 Symposium (HCS), 2015 IEEE. IEEE, 2015, pp. 1–38.

[5] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun et al.,

"Dadiannao: A machine-learning supercomputer," in Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2014, pp. 609– 622.

[6] S. I. Venieris and C.-S. Bouganis, "fpgaconvnet: A framework for mapping convolutional neural networks on fpgas," in Field-Programmable Custom Computing Machines (FCCM), 2016 IEEE 24th Annual International Symposium on. IEEE, 2016, pp. 40–47.

[7] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2015, pp. 161–170.

[8] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 1, pp. 35–47, 2018.

[9] D. Patel, R. Parmar, A. Desai, and S. Sheth, "Gesture recognition using fpga and ov7670 camera," in Inventive Systems and Control (ICISC), 2017 International Conference on. IEEE, 2017, pp. 1–4.

[10] L. Suriano, A. Rodriguez, K. Desnos, M. Pelcat, and E. de la Torre, "Analysis of a heterogeneous multi-core, multi-hw-acceleratorbased system designed using preesm and sdsoc," in Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2017 12th International Symposium on. IEEE, 2017, pp. 1–7.

[11] Y. Bengio et al., "Learning deep architectures for ai," Foundations and trends R in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009.

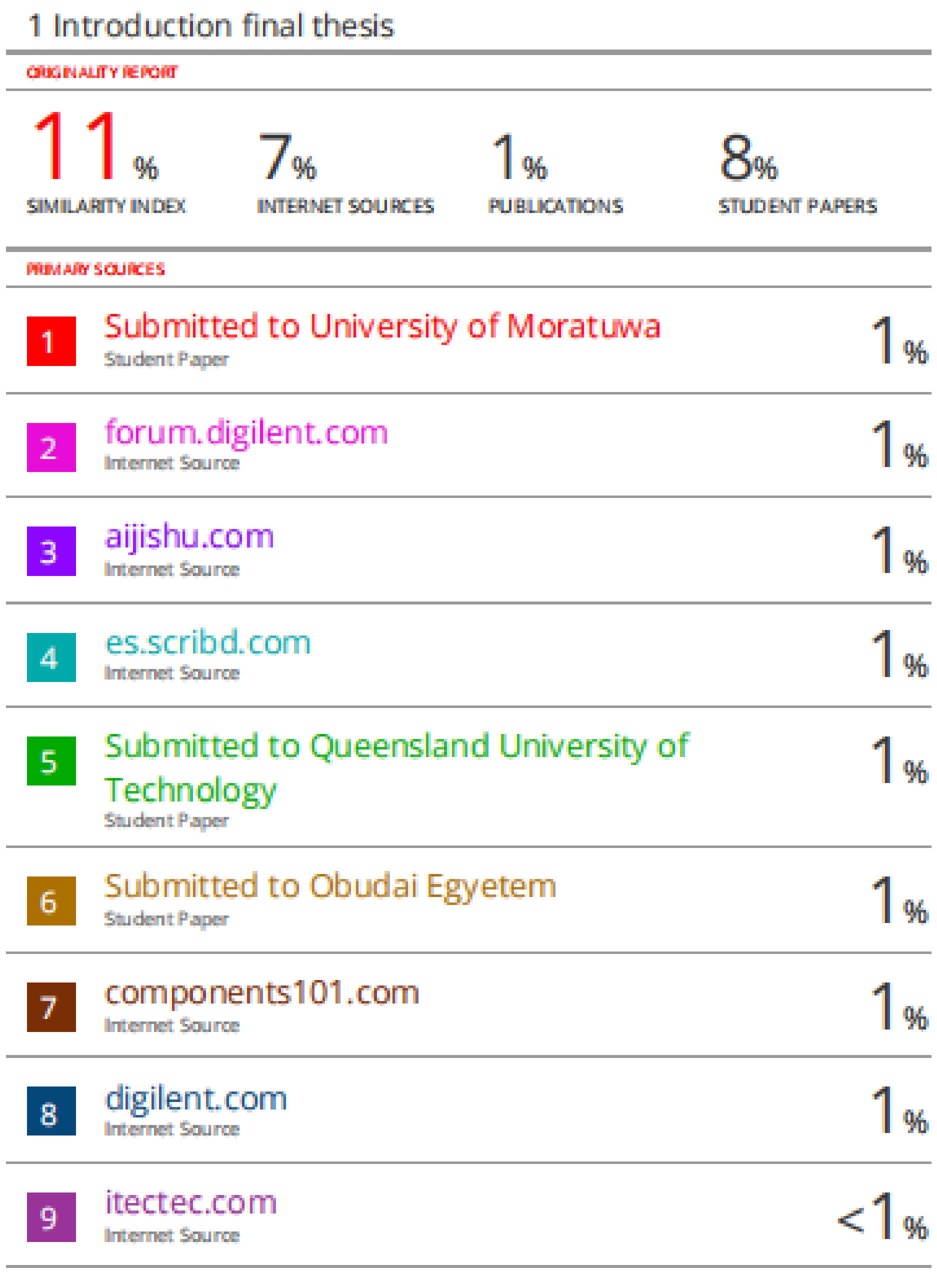
[12] F. Huang, "The application of multi-layer perception networks in the parameters optimization of stamping forming process," in Computer Science and Software Engineering, 2008 International Conference on, vol. 4. IEEE, 2008, pp. 882–886.

[13] D. Lee and H. Yeo, "A study on the rear-end collision warning system by considering neural network," in Intelligent Vehicles Symposium (IV), 2015 IEEE. IEEE, 2015, pp. 24–30.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

[15] K. Srijongkon, R. Duangsoithong, N. Jindapetch, M. Ikura, and S. Chumpol, "Sdsoc based development of vehicle counting system using adaptive background method," in Micro and Nanoelectronics (RSM), 2017 IEEE Regional Symposium on. IEEE, 2017, pp. 235– 238.

Plagiarism Report



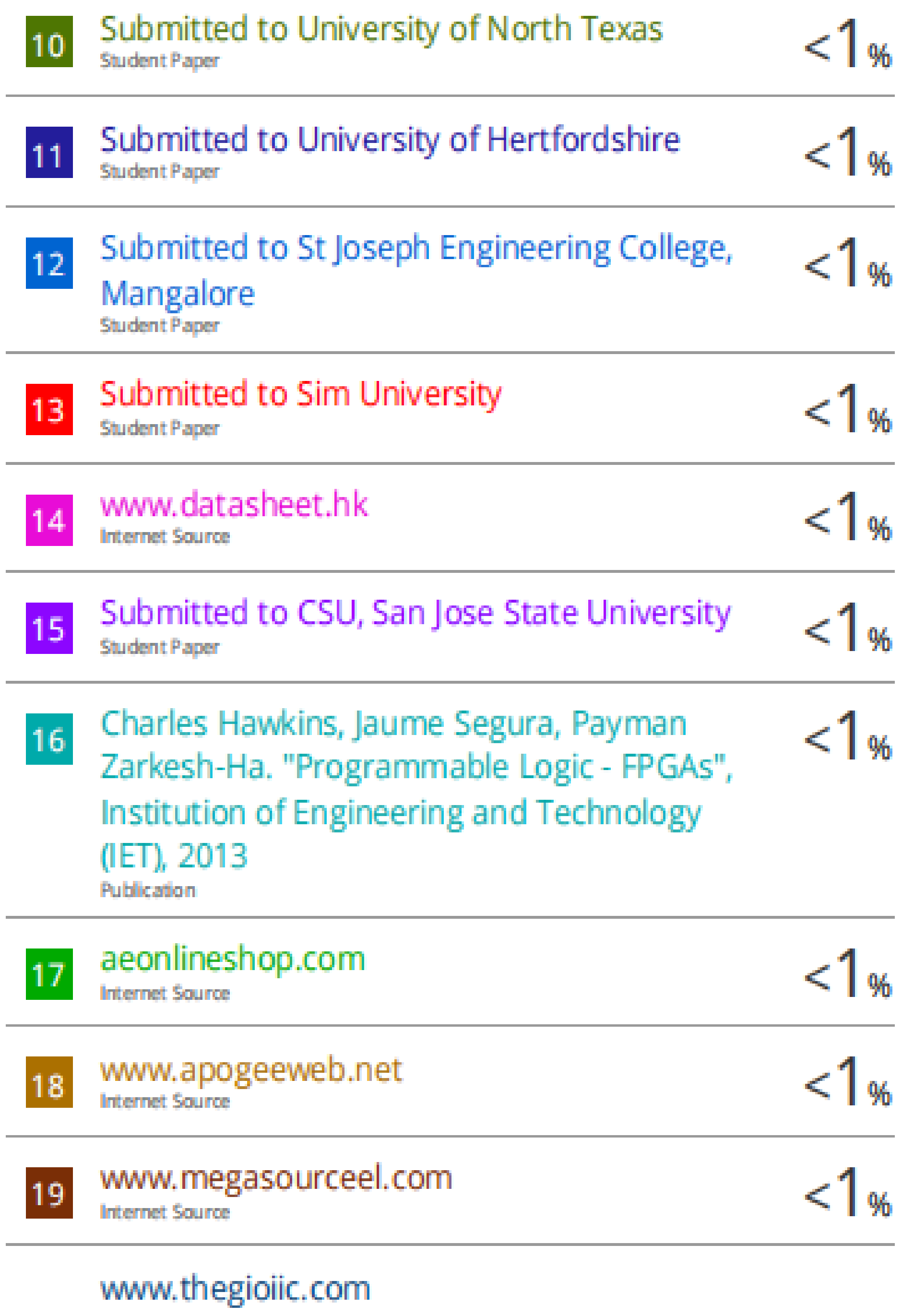
1 Introduction final thesis

ORIGINALITY REPORT

11%	7%	1%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Moratuwa Student Paper	1%
2	forum.digilent.com Internet Source	1%
3	aijishu.com Internet Source	1%
4	es.scribd.com Internet Source	1%
5	Submitted to Queensland University of Technology Student Paper	1%
6	Submitted to Obudai Egyetem Student Paper	1%
7	components101.com Internet Source	1%
8	digilent.com Internet Source	1%
9	itectec.com Internet Source	<1%



10	Submitted to University of North Texas Student Paper	<1%
11	Submitted to University of Hertfordshire Student Paper	<1%
12	Submitted to St Joseph Engineering College, Mangalore Student Paper	<1%
13	Submitted to Sim University Student Paper	<1%
14	www.datasheet.hk Internet Source	<1%
15	Submitted to CSU, San Jose State University Student Paper	<1%
16	Charles Hawkins, Jaume Segura, Payman Zarkesh-Ha. "Programmable Logic - FPGAs", Institution of Engineering and Technology (IET), 2013 Publication	<1%
17	aeonlineshop.com Internet Source	<1%
18	www.apogeeweb.net Internet Source	<1%
19	www.megasourceel.com Internet Source	<1%
	www.thegioiic.com	